UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/654,115	08/30/2000	Dean A. Klein	500050.01	5616

27076 7590 05/10/2006

DORSEY & WHITNEY LLP
INTELLECTUAL PROPERTY DEPARTMENT
SUITE 3400
1420 FIFTH AVENUE
SEATTLE, WA 98101

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 05/10/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/654,115
Filing Date: August 30, 2000
Appellant(s): KLEIN, DEAN A.

Edward B. Bulchis
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 2/13/2006 appealing from the Office action mailed 6/6/2005.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

No amendment after final has been filed.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,115,809	Mattson, Jr. et al.	9-2000
5,765,037	Morrison et al.	6-1998
6,272,599	Prasanna	8-2001
20020078268	Lasserre	6-2002

5,721,893 Holler et al. 2-1998

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

a. Claims 25-27, 29, 32-36, 38-40, 46-50, 52-54 and 60-61 are rejected under 35 U.S.C. 102(e) as being anticipated by Mattson, Jr. et al., USPN: 6,115,809 (hereinafter Mattson).

b. Claims 1-4, 6-8, 13-15, and 17-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and in view of Morrison et al., USPN: 5,765,037 (hereinafter Morrison).

c. Claims 5, 12, and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Morrison et al., USPN: 5,765,037, as applied to claims 1, 15 above, and further in view of Prasanna, USPN: 6,272,599 (hereinafter Prasanna).

d. Claims 9-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Morrison et al., USPN: 5,765,037, as applied to claim 1 above, and further in view of Lasserre, US Pub. No: 2002/0078268 (hereinafter Lasserre).

e. Claims 28, 37, 45, and 51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, as applied to claim 25, 34, 48 above, in view of Prasanna, USPN: 6,272,599.

f. Claims 30-31, 41-44, and 55-58 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, as applied to claim 25, 34, and 48 above, in view of Lasserre, US Pub. No: 2002/0078268.

g. Claim 59 is rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Lasserre, US Pub. No: 2002/0078268, as applied to claim 58 above, and further in view of Prasanna, USPN: 6,272,599.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paras of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the Appellant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the Appellant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.
>
> Note: 35 U.S.C. § 102(e), as revised by the AIPA and H.R. 2215, applies to all qualifying references, except when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. For such patents, the prior art date is determined under 35 U.S.C. § 102(e) as it existed prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. § 102(e)).

2. Claims 25-27, 29, 32-36, 38-40, 46-50, 52-54 and 60-61 are rejected under 35 U.S.C. 102(e) as being anticipated by Mattson, Jr. et al., USPN: 6,115,809 (hereinafter Mattson).

As per claim 25, Mattson discloses a system for determining which portions of a program code to cache and which to not cache, comprising:

a memory device containing a program code (e.g. Fig. 3-4; col. 5, lines 25-56; see col. 3, lines 46-51– Note: code organized in basic blocks for the sake of analyzing branch instructions likelihood reads on code stored in memory, this storage being inherent); and

a processor (e.g. *translate ... into* – Fig. 3, box 29) connected to the memory device, the processor being adapted to be controlled by the program code to direct selected portions of the program code to a cache (e.g. *static code cache, dynamic code cache* - Fig. 3; col. 4, line 59 to col. 5, line 1) based at least in part on cacheability determinations made during compilation (e.g. *compiled, recompiled* - col. 5 line 1-3; *compiler option, Holler* – col. 6, lines 18-29; *at compile time* - col. 5, lines 30-34; col. 6, lines 18-45; *once at compile time* – col. 6, lines 59-62; col. 9, lines 17-22-- Note: Title and Abstract by Mattson read on a context of a compile time determination) of a computer program (Note: Holler as in hereinafter is relied upon only as evidence that certain features are inherent in Mattson's disclosed system).

As per claims 26 and 27, Mattson discloses that the information comprises instructions (e.g. Fig. 1) and data accessed by the computer program (e.g. Fig. 1).

As per claim 29, Mattson discloses memory connected to processor via bus circuitry (Note: bus circuitry connecting processor to memory or peripheral device is inherently disclosed – if necessary see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1).

As per claims 32 and 33, Mattson further discloses a memory device comprising a main memory and an external storage device connected to the processor via the bus (Note: bus circuitry connecting processor to fast memory or peripheral device or main memory is inherently disclosed – if necessary see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1 –Note: Holler is relied upon only as evidence that certain features are inherent in Mattson's disclosed system).

As per claim 34, Mattson discloses a method for controlling the cacheability of information in a computer system, comprising:

compiling a computer program, by making cacheability determination for information associated with the computer program (e.g. e.g. *static code cache, dynamic code cache* - Fig. 3; see col. 6, lines 18-33 or *Holler*, Fig. 4); and

marking at least selected portions of the information according to the determinations (e.g. Fig. 4 - Note: profile-based determination, i.e. *profiled as* 'strong' of 'weak' – reads on marking for appropriate cache storage action; *flag* - col. 9, lines 23-46);

executing the computer program on a computer system, the computer system including cache circuitry (Fig. 3);

detecting the marking of the selected portions of the information during execution of the computer program (e.g. col. 9, lines 1-22); and

directing the selected portions of the information to the cache circuitry according to the marking (e.g. col. 9, lines 1-22; 47-51; Fig. 3-4).

As per claims 35 and 36, refer to rejections 26 and 27 respectively.

As per claims 38 and 39, Mattson further discloses compiling comprises translating source code of the computer program into object code; and programming object code into program directly (e.g. col. 6, lines 18-45; Fig. 3- Note: block partitioning in conjunction with branch prediction heuristics --see col. 6, lines 18-33 or *Holler*, Fig. 4-- before gathering profiling data is reads on compiling including partitioning and heuristics-based instrumentation, i.e. generate an object code leading to an executable to provide subsequent profile-based optimization)

As per claim 40, Mattson further discloses cacheability determinations as to whether the selected portions are cacheable (e.g. Fig. 3).

As per claim 46, Mattson further discloses that the cacheability determination is making determination for a piece of information based on frequency that it will be accessed by the processor during execution of the program (e.g. col. 7, lines 15-44).

As per claim 47, Mattson further discloses a compiler to optimize cacheability determination for a piece of information based on what other piece of information is likely to be overwritten if the first piece is cached (e.g. col. 3, lines 46-67 – Note: branch prediction incorporated by reference discloses concern of overwriting a cache entry).

As per claim 48, Mattson discloses a method for compiling a computer program, comprising:

making cacheability determinations for information associated with the program (e.g. *static code cache, dynamic code cache* - Fig. 3; see col. 6, lines 18-33 or Holler, Fig. 4); and

marking at least selected portions of the information according to the determinations (e.g. Fig. 4 - Note: profile-based determination, i.e. *profiled as* 'strong' of 'weak' -is reads on marking for appropriate cache storage action; *flag* - col. 9, lines 23-46).

As per claims 49 and 50, see claims 26-27 respectively.

As per claims 52 and 53, see rejection of claims 38-39 respectively.

As per claim 54, see rejection of claim 40.

As per claims 60 and 61, these claims correspond to claims 46 and 47 above, respectively, hence are rejected herein using the corresponding ground of rejection as set forth therein.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-4, 6-8, 13-15, and 17-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and in view of Morrison et al., USPN: 5,765,037 (hereinafter Morrison).

As per claim 1, Mattson discloses a computer system having cache circuitry, the computer system adapted to be controlled by a computer program to cache information, comprising:

cache circuitry, including a cache memory to store computer program information; a main memory (e.g. col. 5, lines 25-56; *incorporated by reference* - col. 3, lines 46-51: see *Holler*, USPN: 5721893 -> Fig. 1);

a processor adapted to be controlled by the computer program (e.g. col. 4, lines 24-35 or see *Shah*, USPN: 6,205,545 -> col. 4, lines 28-50 – Note: a processor executing a program inherently signifies its being controlled by the program) and to direct selected portions of the information to the cache circuitry based at least in part on cacheability determinations (e.g. Fig. 3) made during compilation of the computer program (e.g. *compiler option, Holler , at compile time*- col. 5, lines 30-34; col. 6, lines 18-45; *once at compile time* – col. 6, lines 59-62); and

bus circuitry, connecting the processor, the cache circuitry, and the main memory (see col. 3, lines 46-51 or *Holler*, USPN: 5721893 -> Fig. 1– Note: bus circuitry is inherent to any computer system).

But Mattson does not explicitly mention directing selected portions of the program information to the cache circuitry via cooperation with a bus interface unit. With the so-disclosed hardware components by Mattson (e.g. see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1), one skill in the art would recognize the obvious existence of an interface unit to communicate the cache circuitry and the bus system as taught by Mattson via Holler. Further, in a system to route instructions data to the appropriate hardware, e.g. cache, using compilation information (see Morrison: col. 33, lines 16-23) analogous to Mattson's system to optimize execution time for branch operations via profiling information prior to caching execution instructions/data, Morrison discloses the use of bus interface unit (unit 1544 – Fig. 15) to cooperate with runtime translating address and to direct instructions to cache (e.g. col. 27, line 56 to col. 28, line 31) via using algorithms establishing weighted basic blocks (col. 24 lines 26-63). Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a bus interface unit as taught by Morrison to Mattson's system, in case the latter does not already include one such unit, because this would allow the intermediate step of spatially and logically re-directing or selectively dispatching of data to the correct storage place, cache or buffers, enhancing fault-free transmission of cacheable data via bus (Morrison: col. 30, lines 4-17).

As per claims 2 and 3, these claims correspond to claims 26 and 27 above, respectively, hence are rejected herein using the corresponding ground of rejection as set forth therein.

As per claim 4, Mattson further discloses that selected portions are marked during the compilation of the program such that the bus interface unit can identify the selected portions during execution of the program (e.g. col. 6, lines 18-45; Fig. 3- Note: block partitioning in conjunction with branch prediction heuristics --see col. 6, lines 18-33 or Holler, Fig. 4-- for profiling data based on compiler instrumentation or rule-based heuristics is reads on marking during compilation for information capture leading to identifying of marked portions for cache storage during the execution of the optimized and recompiled program – see col. 9, 1-51; the identifying by the BIU being obvious as per the rationale as set forth in claim 1).

As per claims 6 and 7, these claims correspond to claims 38 and 39 above, respectively, hence are rejected herein using the corresponding ground of rejection as set forth therein.

As per claim 8, Mattson further discloses that cacheability determinations comprise determining that the selected portions are cacheable (e.g. Fig. 3; see col. 6, lines 18-33 or Holler, Fig. 4).

As per claim 13, Mattson further discloses a compiler to optimize cacheability determination (e.g. col. 6, lines 18-45; Fig. 3- Note: block partitioning in conjunction with branch prediction heuristics --see col. 6, lines 18-33 or Holler, Fig. 4-- for gathering profiling data is reads on compiler operable for including partitioning and heuristics-based instrumentation leading to profile-based determining on cacheability of identified portions in the final optimized code).

As per claim 14, Mattson discloses cacheability determination for a first piece of information is based at least in part on whether caching of the first piece of information is likely not to cause cache mis-prediction or potential cache miss, all of which under the scheme of

seeking improved execution efficiency and cache management (e.g. col. 1, lines 41 to col. 3, line 67) but does not specify whether such caching is likely to cause thrashing of the cache circuitry. The problem of cache being limited leading subsequent techniques to manage cache resources to avoid thrashing was a known concept in the art at the time the invention was made. Morrison for example, in the system to optimize the execution of programs using a bus and cache circuitry associated with profile information analogous to that of Mattson, discloses parameters (*tag, IFT, LPN*- col. 31, lines 24-64) based on which to process execution and prevent cache miss (e.g. unit 1640-Fig. 16; col. 32, lines 16-44 – Note: cache miss avoidance is reads on minimizing cache thrashing). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement parameters determined at compile as taught by Mattson in order to prevent cache thrashing issue as suggested by Morrison because the intent to improve processor latency as suggested by Mattson necessarily implies alleviating memory issues such as cache misses and recovery, i.e. thrashing, which entails processing downtime.

As per claim 15, Mattson does not explicitly specify a cache management scheme but discloses a scheme to prevent cache mis-prediction or potential cache miss, all of which under the scheme of seeking improved execution efficiency or cache management (e.g. col. 1, lines 41 to col. 3, line 67) while Morrison suggests a management scheme in association with the hardware used to process instruction stream, bus and cache circuitry and an cache control structure(e.g. manage -col. 6, lines 1-22; Fig 15-16). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a cache management scheme in the cache and I/O circuitry as suggested by Morrison to Mattson's system to process instruction data during execution and base the cacheability determination thereupon because the

more complicated the system is in terms of bus size, memory and instructions set architecture, the more management is needed in coordinating data from their source storage, e.g. memory, registers, to their destination, e.g. cache, via bus circuitry; such coordination being routing, and/or synchronization of instructions stream as mentioned by Morrison.

As per claim 17, Mattson further discloses a compiler to optimize cacheability determination for a piece of information based on frequency that it will be accessed by the processor at execution (e.g. col. 7, lines 11-44).

As per claim 18, Mattson further discloses a compiler to optimize cacheability determination for a piece of information based on what other piece of information is likely to be overwritten if the first piece is cached (e.g. col. 3, lines 46-67 – Note: branch prediction incorporated by reference discloses concern of overwriting a cache entry).

As per claim 19, Mattson further discloses that determinations are accomplished during compilation into object code utilizing profile-based optimizations (e.g. col. 6, lines 21-45).

As per claims 20 and 21, Mattson discloses a system controller, adapted to send and retrieve instructions; and a bus device connecting an external device to the bus (refer to rejection of claims 32 and 33 from above).

As per claim 22, Mattson discloses an external device providing instructions utilized by the processor for optimization (e.g. see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1– Note: memory/media storage storing profile data is reads on providing stored instructions for optimization/execution).

As per claims 23 and 24, Mattson further discloses that instructions are compiled by a compiler to optimize cacheability determinations (e.g. . *compiler option, Holler , at compile*

time- col. 6, lines 18-45; *once at compile time* – col. 6, lines 59-62; Fig. 3- Note: block partitioning in conjunction with branch prediction heuristics --see col. 6, lines 18-33 or *Holler*, Fig. 4-- before gathering profiling data is reads on compiling including partitioning and heuristics-based instrumentation for cacheability determination) and that cache circuitry and processor are provided on one single chip (col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1).

5. Claims 5, 12, and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Morrison et al., USPN: 5,765,037, as applied to claims 1, 15 above, and further in view of Prasanna, USPN: 6,272,599 (hereinafter Prasanna).

As per claim 5, Mattson discloses using compiler including analysis information to determine cacheability of program data and marking of instructions for appropriate storing in cache (Fig. 3-4), or selectively caching during execution based on compiling information generated from partitioning or heuristics-based instrumenting of instructions (see col. 6, lines 18-33 or Holler, Fig. 4); but fails to disclose that such compiler-generated information contains marking bits, and that the compiler sets the marking bits of the selected portions of the information during compilation. As another approach to Mattson's compiler-based insertion of instrumentation instructions as mentioned above, Prasanna, in a system to improve execution time by alleviating cache thrashing using compilation information analogous to the compile time determining of instructions caching as taught by Mattson, discloses the compiler marking of bits in instructions datum, e.g. ILP, in order to identify which lines to cache or not to cache in order to avoid interferences patterns (*cache/no-cache bit* – col. 2, lines 29-58; Fig. 1-2). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add the use of marking bits to instructions datum by the compiler as suggested by Prasanna to

Mattson 's method of determining of target address for execution data access optimization because this would further speed up the translating of address information into cached data in the execution of the optimized code as taught by Mattson, to thereby reduce cache thrashing as also suggested by Prasanna (col. 1, lines 48-67).

As per claim 12, Mattson combined with Morrison does not specify that the cache-circuitry includes at least one N-way associative cache with N>1; but Morrison suggests a speed increase in fetching instructions from cache should a fully-associative memory be used (e.g. col. 31, lines 24-63). Further, Prasanna in the system to mark bits in instruction datum at compile time for cacheability determination as mentioned above, discloses a N-way associative cache (col. 3, lines 1-14). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the cache circuitry in the memory latency optimization scheme as taught by the combination Morrison/Mattson in a way that a more-than-one-way associative type of cache as taught by Prasanna because according to Prasanna, this would improve performance of a computer when the size of the sequential address stream to process is greater than the total cache memory available.

As per claim 16, Mattson in view of Morrison discloses a cache management scheme as addressed in claim 5 above; but fails to disclose such cache management scheme comprises the level of associativity of cache memory. But in view of Prasanna in the disclosing of a N-associative cache scheme as mentioned in claim 12 above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to incorporate in the cache management scheme as suggested by Mattson/Morrison the level of associativity of cache as

taught by Prasanna because of the same benefit mentioned above in the rejection set forth in claim 12.

6. Claims 9-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Morrison et al., USPN: 5,765,037, as applied to claim 1 above, and further in view of Lasserre, US Pub. No: 2002/0078268 (hereinafter Lasserre).

As per claim 9, Mattson discloses cacheability determinations as to whether to cache selected portions in the program but does not specify that the cache circuitry includes first cache memory and second cache memory; nor does Mattson disclose said determinations comprise determinations as to cache said selected portions in the first or second cache memory. However, Mattson suggests 2 types of cache, a dynamic type for hard to predict branch instructions and a static type for easy to predict instructions (Fig. 3,5), hence has disclosed a cache level where fast fetching therefrom can be done and a cache level where slower fetching is likely. Morrison, in a system as mentioned in claim 1 to cache data using compilation information analogous to Mattson's method, discloses the several partitions of cache to store instructions (e.g. Fig. 15; col. 28, lines 15-31). Further, Lasserre, in a system analogous to Mattson and Morrison' s, using cache circuitry and pre-configured memory information to determine dynamic cache storage to alleviate cache conflicts, discloses level 1 and level 2 caches with different size and application each (p. 3, paragraghs 0048, 0050; cache *113, 114*-Fig. 1). In view of the teachings and suggestions by Mattson/Morrison and Lasserre's suggestion of different size caches and purposes, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement 2 level caches as suggested therefrom and combine the cache determination and partitioning as suggested by Mattson (and further enhanced by Morrison) with

the use of 2 level caches to determine as to which cache the portions selected by compilation in Mattson's method should be stored. One of ordinary skill in the art would be motivated to do so because directing a certain size instruction or data to be cached in an appropriate size cache partition as suggested by Lasserre would enhance resource usage efficiency and also in terms of memory spatial and temporal optimization.

As per claim 10, Mattson does not teach the first level cache and the second level cache; but in view of Morrison and Lasserre's teachings in claim 9 above, this limitation would have been obvious herein using the same rationale set forth therein.

As per claim 11, Mattson does not specify that the cache circuitry supports write-back and write-through caching and that cache determinations comprise the write-back or write-through method. Official notice is taken that in techniques used for cache management such as taught by Mattson, the write-back and write-through techniques were well-known practices at the time the invention was made, so as to avert runtime memory from being bogged down with extraneous read and write operations. Lasserre, in a system to alleviate cache conflict using memory information to set cacheability for execution, discloses both write-back and write-through caching methods (e.g. p. 6, paragragh 0086). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a cache system as suggested by Lasserre to that disclosed by Mattson (and further enhanced by Morrison) because these caching methods are known to support selective reduction of latency with regard to the temporary or stable state of the data being used during execution, thus enhancing further execution time efficiency.

7. Claims 28, 37, 45, and 51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, as applied to claim 25, 34, 48 above, in view of Prasanna, USPN: 6,272,599.

As per claim 28, Mattson discloses using compilation analysis information to determine cacheability of address data (Fig. 3-4), or selectively caching during execution based on compiling information generated from partitioning or heuristics-based instrumenting of instructions (see col. 6, lines 18-33 or *Holler*, Fig. 4); but fails to disclose that such information contains marking bits, and that the compiler sets the marking bits of the selected portions of the information during compilation. Prasanna, in a system to improve execution time by alleviating cache thrashing as mentioned in claim 5 above, discloses the compiler marking of bits in instructions datum, e.g ILP, in order to identify which lines to cache or not to cache in order to avoid interferences patterns (*cache/no-cache bit* – col. 2, lines 29-58; Fig. 1-2). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add the use of marking bits to instructions datum by the compiler as suggested by Prasanna to Mattson 's method of determining of target address for execution data access optimization for the same rationale as set forth in the rejection of claim 5 above.

As per claim 37, Mattson fails to disclose that each piece of information contains marking bits and that such marking includes setting the marking bits of at least the selected portions of the information. In view of the marking of bits in the instructions datum as taught by Prasanna in claim 28 or 5 above, this limitation would also have been obvious and is herein rejected with the same rationale as set forth in the rejection of claim 5 above.

As per claim 45, Mattson discloses at least one level of cache memory (Fig. 3) but fails to disclose that the act of making cacheability determinations for program information is based on a level of associativity of cache memory. But in view of Prasanna in the disclosing of a N-associative cache scheme as mentioned in claim 12 above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to incorporate in the cache circuitry as suggested by Mattson at least one level of associativity of cache as taught by Prasanna because of the same benefit mentioned above in the rejection set forth in claim 12.

As per claim 51, this claim includes the same limitation as claim 37 above, hence is rejected herein using the same rationale as set forth therein.

8. Claims 30-31, 41-44, and 55-58 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, as applied to claim 25, 34, and 48 above, in view of Lasserre, US Pub. No: 2002/0078268.

As per claims 30 and 31, Mattson discloses a cache circuitry connected to processor (Fig. 3) but does not explicitly disclose (re claim 30) a level one cache; nor does Mattson disclose (re claim 31) a level two cache connected to the processor and memory via bus circuitry. But in view of the teachings by Lasserre as mentioned in claim 9 above (p. 3, paragraghs 0048, 0050; cache *113, 114*-Fig. 1), these level one and level two limitations would have been obvious for the same rationale as set forth in claim 9 above.

As per claim 41, Mattson does not explicitly disclose that the cache circuitry includes first cache memory and second cache memory; nor does Mattson disclose said determinations comprise determinations as to cache said selected portions in the first or second cache memory. However, Mattson suggests 2 types of cache, a dynamic type for hard to predict branch

instructions, and a static type for easy to predict instructions (Fig. 3,5), hence has disclosed a cache level where fast fetching therefrom can be done and a cache level where slower fetching is likely. Lasserre's use of 2 level caches (p. 3, para 0048, 0050; cache *113, 114*-Fig. 1) has been mentioned in claim 9 above. Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement 2 level caches as suggested by Lasserre and combine the cache determination as suggested by Mattson (and further enhanced by Morrison) with those 2 level caches to determine as to which cache level the portions selected by compilation in Mattson's method should be stored. One of ordinary skill in the art would be motivated to do so because of the same benefits as set forth in rejection of claim 9 above.

As per claim 42, Mattson does not specify both write-back and write-through caching methods, nor does Mattson disclose determining whether to cache selected portions using those 2 methods. This claim limitations have been addressed using Lasserre's teachings (e.g. p. 6, para 0086) in claim 11 above, hence is rejected herein using the same grounds of rejection as set forth therein.

As per claim 43, Mattson discloses cacheability determination for a first piece of information is based at least in part on whether caching of the first piece of information is likely not to cause processor to have a mis-prediction conflict (e.g. e.g. col. 1, lines 41 to col. 3, line 67), but does not specify whether such caching is likely to cause thrashing of the cache circuitry. The problem of cache being limited leading subsequent techniques to manage cache resources to avoid thrashing was a known concept in the art at the time the invention was made. Lasserre, in a system to alleviate cache conflict using memory information to set cacheability of data for execution analogous to that of Mattson, discloses cache circuitry handling cache miss/hit logic

and clean-up process as a result of cache miss (e.g. *Hit/Miss logic 510* - Fig. 4; p. 4, 5, 6 - paras 0062, 0063, 0074). By cache miss handling and recovering, one skilled in the art would recognize the cache thrashing issue as claimed. In view of the above known concepts and recognitions, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement information determined at compile as taught by Mattson in order to prevent cache thrashing issue as suggested by Lasserre because the intent to improve processor latency as suggested by Mattson necessarily implies alleviating memory issues such as cache misses and recovery, i.e. thrashing, which entails processor downtime.

As per claim 44, Mattson does not specify a cache management scheme but discloses a scheme to store instruction data in memory and an I/O controller (e.g. col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1) while Lasserre suggests a management of cache in association with the hardware used to process instruction stream (e.g. *Hit/Miss logic 510, cache control 530* - Fig. 4; p.6 – para 0074). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a cache management scheme via the cache control and hit/miss structure as suggested by Lasserre to Mattson's system to process instruction data during execution and base the cacheability determination thereupon because the more complicated the system is in terms of bus size, memory and instructions set architecture, the more management is needed in coordinating data from their source to their destination; such coordination and flow control for optimizing the performance on specialized processors instructions set with numeric intensive execution as suggested by Lasserre (col. 1, paras 0004, 0005).

As per claim 55, Mattson does not disclose that cacheability determinations comprise determinations as to cache said selected portions in the first or second cache memory. But in view of the teachings of Lasserre as mentioned in claim 41 above, this claim is rejected using the same rationale as set forth therein.

As per claim 56, this claim includes the limitation as whether to cache using write-back or write-through method as in claim 42 above, hence is rejected using the corresponding rationale set forth therein.

As per claims 57 and 58, these claims correspond to claims 43 and 44 above, respectively, hence are rejected herein using the corresponding ground of rejection as set forth therein.

9. Claim 59 is rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Lasserre, US Pub. No: 2002/0078268, as applied to claim 58 above, and further in view of Prasanna, USPN: 6,272,599.

As per claim 59, Mattson combined with the teachings of Lasserre, fails to disclose that the cache management scheme comprises the level of associativity of the cache. But Prasanna, in the invention as mentioned in claim 45 above, discloses a N-way associative cache memory. In combination with the cache circuitry and cache management scheme of Mattson/Lasserre (re claim 44), Prasanna's teachings would have help render the above limitation obvious for the same rationale as set forth in claim 45 above.

(10) Response to Argument

10. Appellant's arguments filed 2/13/2006 have been fully considered but they are not persuasive. Following are the observations in regard thereto.

Anticipation by Mattson

(A)

(i) Appellant has submitted that Mattson's prediction method using same physical pages of memory called 'static code caches' and 'dynamic code caches' has nothing to do with a caching system as that term has been used by Applicant; such that Mattson's thus mentioned caches are simply different pages of the same physical memory, like contiguous locations of a memory (Appl. Arguments, middle pg. 7); and that Mattson's strong and weak branch instructions are stored in different pages of the same memory, which are identified by prediction flags (Appl. Arguments, pg. 8, top).

First, it is noted that the claims only recite a memory and directing instructions to a cache, and do not recite that such recited memory and cache are physically distinct, as asserted above. For example, claim 25 only recites 'direct selected portions of the program code to a cache based on ... cacheability determinations'. Mattson's disclosed compiler directs some instructions to a 'static' code block or 'dynamic' cache are based on determination as to whether these instructions are determined statically to be either weak or strong branch instructions, respectively (see Mattson's col. 8, lines 13-54). Mattson's determination and segregation of branch instructions is done at compile time (see col. 6, lines 46-62); and based on the algorithmic approach to partition code block from Figure 4-5, specific branch instructions can be partitioned appropriately in the 'static' or 'dynamic' cache areas. This teaching reads on 'direct selected portions of the program code to a cache based in part on cacheability determinations made during compilation of a computer program'.

(ii) Next, Appellant emphasizes on the fact the Mattson's memory pages are not cache and that the instant invention clearly entails the existence of a cache being distinct from the program code memory. Since the claim language does not require that the memory and cache are physically separate from each other, making compile time determination to move specific instructions in memory code to area called 'static' or 'dynamic' cache as disclosed in Mattson has therefore met the limitation. The recitation of 'direct ... program code to a cache' does not preclude any generic *memory pages* from being called cache areas. Further, the term 'cacheability determination' is not clearly defined in the claim so to preclude determination whether to store branch instructions in 'static' or 'dynamic' cache as disclosed in Mattson. Thus, Appellant's arguments amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

(B) Appellant has submitted that Mattson's cited portions in the Final Office Action do not teach a memory device or even mentions the word memory. In response, it is again necessary to point teaching considered known in prior art; and this would be cited not only in Mattson's (Fig. 1, Fig. 4; col. 8, lines 18-24) but also in works incorporated by references as in Holler (see Fig. 1 Memory 13, Cache 12) and Shah (basic block code – Fig. 1; address - col. 5, lines 36-49); and the very concept behind storing of code blocks as above cited --for analysis via profiling or predicting -- would strongly imply storage medium for such code. As set forth in the rejection, it has been pointed out that Holler is relied upon as evidence of inherent subject matter. That is, Mattson refers to a specific HP computer architecture (e.g. col. 2, lines 49-56; col. 4, lines 54-63; Fig. 2), which is disclosed in Holler, therefore Mattson's disclosure includes the specific

architecture and associated hardware/structures as claimed. The memory to store code being organized in basic blocks (see Mattson's Fig. 4) so to enable partitioning or profiling based analysis the memory to contain the basic blocks is considered inherent (emphasis added) in Holler or in Mattson. As for using multiple reference to put forth a 102 rejection, refer to MPEP § 2131.01, based upon which the use of memory to support code being divided in basic blocks for a compile time analysis and/or being loaded to execute profiling process would enable the common endeavor by Mattson's and also dictates that without memory storage, the code cannot be executed to yield such profile data, let alone being analyzed for partitioning.

(C) Appellant has submitted that anticipation by Mattson cannot be combined with Holler (Appl. Arguments pg. 8, bottom). It is noted that Mattson's disclosure has incorporated Holler as references being part on Mattson's original disclosure (see Mattson *incorporated by reference*: col. 1, lines 13-18; col. 3, lines 46-67) and all the teaching by Holler will become prior art disclosed by Mattson, thus coming from one reference. Besides, the reference by Holler is aimed to show that code storage for analysis by a compiler is stored in memory; and this is reading on the memory code limitation of claim 25. There would be no need to cite Holler for an alleged combination with Mattson's purported for a rationale for obviousness because Mattson's original disclosure has incorporated Holler's specifications as part of Mattson's. As per combining multiple references in one art, the inherency of memory in Mattson's method has been discussed in part in section B above.

(D) Appellant has submitted that there no citations by Mattson that disclose a processor directing selected instructions stored in a memory and that Holler does not describe 'cacheability determination' in view of Holler's teaching (Appl. Arguments pg. 9, middle). Holler has been

cited to support a known concept to store code in memory for compiling or making profiling-based determination on branch prediction of basic blocks; while Mattson's determination techniques have been cited to meet what is recited as 'cacheability determination' at compile time, which has been put forth in the Office Action in light of the above section A. The mere fact that Mattson discloses *profile* in *program trace* entails execution of code being loaded in memory for profile data to be used for further analysis (see step 26, Fig. 3). As set forth in the rejection, it has been pointed out that Holler is relied upon as evidence of inherent subject matter, e.g. memory in Hewlett Packard computer architecture.

(E) Appellant has submitted that Mattson's storing of instructions in a same memory and Holler's buckets of a same cache do not amount to disclosing the limitation of claim 25 (Appl. Arguments pg. 10, top para). This argument is referred back to section A above for Mattson's and section D for the argument against Holler.

(F) Appellant has submitted Mattson does not teach or suggest making cacheability determinations during compilation to *control* caching during execution of the instructions (Appl. Arguments, pg. 10, 2nd paragraph).

The limitation such as 'control caching during execution of the instructions' is not recited in the claims. Further, section A has addressed how Mattson has used profiling data via trace of codes at compile time and use such information for partitioning code into cache areas. The claim as recited does not make it explicitly clear that a physical cache is distinctly built within the very system having a memory code and that the processor is actually storing the program code in cache via some determination; that is, a processor *adapted to* direct code *to a cache* does not necessarily enforce that a storing step has taken place and/or that a code cache is physically

existing in the system when this step is implied as being performed, thus at best would be construed as though the final result of the system claim is the determination outcome implying a knowledge about which code will go to cache. Appellant's explanation about branch prediction and cache system (Appl. Arguments, pg. 10, last para to pg. 11, top 2 para) amounts to mere insights that are not commensurate with the scope of the claimed limitations as these have been construed via broad reasonable interpretation. The claim as a whole amounts to a processor adapted to direct program instructions to be cached based on some determination at compile time, with the program code stored in code memory. There is no sufficient teaching in the claim to enforce about the nature of what needs to be cached, what cache amounts to, and whether a storing actually takes place with respect to the system comprising only of a memory and the processor. Part of this subject has been addressed in section A above.

(G) Appellant has disagreed that the claimed invention about marking of selected portions to be cached is 'equivalent' to Mattson's marking of branch instructions (Appl. Arguments, pg. 12, last para, pg. 13, top para).

The rejection has clearly set forth the correspondence between Mattson's marking of branch instructions to the 'detecting the marking of selected portions of the information...' and the 'directing the selected portions ... to the cache circuitry according to the marking'; the equivalence test issue becoming moot because the rejection has mapped the corresponding teaching/features with an explanatory note describing how the cited parts 'reads on' the claimed parts. And when a cited portions reads on a claimed feature, there is anticipation until such anticipation is justifiably rebutted.

(H) As per claims 34 and 48, Appellant has submitted that the Office Action ignores substantial differences between storing cached instructions and branch prediction (Appl. Arguments, pg. 13, bottom).

In response to applicant's argument that the references fail to show these features of applicant's invention, it is noted that the features upon which applicant relies (i.e., differences between caching and branch prediction) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Obviousness rationale combining Mattson and Morrison:

(I) As per claim 1, Appellant has submitted that Mattson's teachings in view of Morrison's bus interface unit would not result in having separate main memory and cache memory (Appl. Arguments, pg. 17, 2nd para).

Examiner has already addressed how the claim fails to clearly recite the disposition of a physical cache memory with respect to the system main memory has been discussed at length in sections A and F above. In claim 1, the cache memory is adapted to store information related to a computer program; the main memory to store the information; and the bus to cooperate with the processor, the main memory and the cache circuitry. According to broad reasonable interpretation of the claim, the processor when executing the program would direct portions of 'the information' to the cache circuitry based on some determinations at compile time. The claim does not recite any specifics as to the information being thus directed, specifics on the path to the cache, or the disposition of the main memory with respect to the cache memory when the act of directing code to cache is perceived as directing 'the information', and that both the cache

circuitry and the main memory are for storing information related to a program and that the bus is generically perceived as for connecting cache, memory and processor. If the endeavor is to allow some program instructions after judicious analysis (Mattson's profiling) to be moved from its original place, like some sections of memory, to other memory sections in order to enable fault-free execution, particularly when that the memory sections wherein these instructions are relocated are called 'cache' (see Mattson: Fig. 3-5), then the cache storing has been disclosed because moving code for the purpose of improving execution time or to obviate delay as contemplated by Mattson, Holler or Shah (incorporated by reference, Shah: USPN 6,205,545 or Holler: USPN 5,721,893) clearly depicts the purpose of managing memory cache so that only certain instructions are stored as compared to the rest of the program. It is therefore clear that Mattson by moving code to specified areas called 'cache' has provided teachings that read on a cache memory for the purpose of expediting execution in an improved fault-minimizing approach; thus hence has met the claim limitation. That is, the argument that Mattson's moving from contiguous memory pages to other pages not fulfilling the subject matter of separate cache and memory as claimed is to be considered not convincing. Next, Mattson's directing of instructions into a block of memory called cache would necessarily require a transportation means to enable such transfer from one place to another, even from among different sections of a big memory entity—like a processor bus; and while this bus circuitry is not explicitly disclosed in Mattson, it would have been an obvious feature to Mattson's process of transferring in view of Morrison's disclosure. As set forth in the rejection, Mattson's directing of particularly identified instructions via profiling analysis (compile time) would be enhanced when combined Morrison's bus circuitry endowed with its dispatch checking functionality; and this is what constitutes a

USC 103(a) case of obviousness based on a combination of teachings for a specific benefit. Thus, this rationale of obviousness has to be rebutted in light of combined teachings, let alone that fact that a bus is always integral (otherwise construed as an inherent feature) to any computer system like that of Mattson's to enable transfer of data between memory and processor. Appellant fails to show specifics as to why combining Morrison with the cache storing by Mattson's would be resulting adverse effects or that the purposes of Mattson and Morrison's invention would not allow that a bus circuitry be installed to Mattson's system, and precisely why so.

(J) Appellant has submitted that caching involves storing cache information in a separate memory; and that cacheability determination is markedly different from making branch prediction (Appl. Arguments, pg. 17, 3rd para).

The argument about differences between branch prediction and cache storing determination has been addressed in sections A, F, and H above. The argument about a cache circuitry and a separate memory has been addressed in section I above.

(K) Appellant has submitted that Morrison's shared static information or *additional pieces of information* do not remotely relate to the cacheability of instructions (Appl. Arguments, pg. 18, middle para) and that Morrison's determination of what to store in the caches are made during execution of the program (Appl. Arguments, pg. 18, last 2 para).

The rejection as set forth in the Action has pointed what feature is deemed not explicitly taught by Mattson; and by using Morrison which presents this feature, has put forth a rationale as to render the missing feature obvious. If Mattson is to be expected to also disclose a bus circuitry, then there would be no point to set up a USC 103 rejection; and if Morrison is to be

expected to disclose all the features of the claim 1, then Morrison would be otherwise applied as an anticipating prior art in a USC 102. The rejection has shown that Morrison discloses a bus circuitry, which Mattson explicitly does not. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). The question is whether the use of a bus circuitry in a process of improving code execution time via a better cache management using information to support such execution would have been obvious. As pointed out in the rejection, both Morrison and Mattson are trying to improve runtime execution by providing techniques via weighted basic blocks (see Morrison: *resultant re-grouping ... basic blocks ... weighted assignment ... branch probabilities* - col. 24 lines 26-63) or via profiling from code trace (see Mattson Fig. 3), the weight or the profile data being collected thus attesting to the fact that data collection is done for compile time analysis. The fact that Morrison loads instructions set to cache to a point where a branch instruction needs to be addressed outside of the execution set (col. 24, lines 56-63) is only reminiscent of Mattson's caching of branch instructions marked for 'dynamic cache', based on which such instructions are likely to be resolved during runtime. The rejection has set forth the benefits of using a bus circuitry as taught by Morrison to impart to Mattson's instructions transfer; and Appellant fail to point out the deficiencies of such rationale.

(L) Appellant's argument that Morrison 'fails to disclose the subject matter of claim 1 that is missing from Mattson, i.e. making cacheability determinations during compilation...' (Appl. Arguments pg. 19, 3rd para)

This would fall under the ambit of the topic being discussed in sections I and K; because if the Morrison's input to the rationale of rejection has to be rebutted, such rebut has to be commensurate with the grounds based on which the USC 103(a) has been put together, not how much Morrison fails to fulfill a limitation that has been construed as already met by the base reference, i.e. "making cacheability determinations during compilation...".. Therefore, the argument that references cannot be combined even in hindsight (Appl. Arguments pg. 19, bottom) is insufficient to overcome the rejection. That is, the Appellant points out teachings that are not related to the scope of the rationale set forth in the USC 103(a) and further fails to clearly establish why the use of the feature (bus circuitry) provided from one reference (Morrison) into the context of the other reference (Mattson) would have been obvious as set forth in the rejection.

For the above reasons, it is believed that the rejections should be sustained.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

Respectfully submitted,

Tuan A Vu

Conferees:

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

TUAN DAM
SUPERVISORY PATENT EXAMINER